CSC110 Fall 2021 Assignment 1: Written Questions

Jamie Yi

September 27, 2021

Part 1: Data and Comprehensions

1. **Imagine this scenario...**

   (a) A list. It stores multiple items so it can store all of my team-mate's notes. Secondly, lists are ordered so it's possible that my teammate sorts her list in ascending priority order. So to get the highest priority task, she could just do todo[0].

   (b) An int. Points aren't fractional in table tennis so a float is not required. Also this value only requires the total number of points played in the game, not the points scored per player so there only needs to be one value to represent this. No need for a list/set.

   (c) A set. We want to catalog the unique types of fruits so a set is perfect for that since it doesn't allow duplicates. Order doesn't matter here so that's another reason why to use the unordered set.

   (d) A Boolean. The question "did I win the game?" can be answered with yes or no. That means it can also be answered with a Boolean true or false.

   (e) A dictionary. For the keys, it can be integers representing the *nth* point scored. The values would be the name of the person who scored the point. So for the first two points as described in the scenario, the dictionary would look like: {1: 'me', 2: 'teammate'}.

2. **Exploring comprehensions.**

   (a) i. ['H', 'e', 'l', 'l', 'o', ' ', 'D', 'a', 'v', 'i', 'd']
   ii. The result is a list, containing strings.

   (b) i. {'D', 'i', 'o', 'e', ' ', 'l', 'H', 'd', 'v', 'a'}
   ii. The result is a set, containing strings.
   iii. Both results contain the same unique elements. However, the set in part b is smaller than the list in part a because the set does not contain any duplicate elements.

   (c) [False, True, False, False, True, False, False, True, False, False, False] This value represents whether or not each character in 'Hello world' is in the set vowels or not. In layman's terms, for each character in 'Hello world', is it a vowel?

   (d) The first in(c **in** vowels) is used to evaluate whether or not the string c is an element in the set vowels. That expression evaluates to a Boolean. The second in(c **in** 'Hello world') is used to tell the variable c to become each iterable in 'Hello world'. In this case, as the comprehension iterates the in tells c to become the next character in the string each iteration.

Part 2: Programming Exercises

Complete this part in the provided `a1_part2.py` starter file. Do **not** include your solution in this file.

Part 3: Pytest Debugging Exercise

1. test_section_average_all_grades_equal passed the test
   test_class_average_no_grades_equal failed the test
   test_class_average_many_students failed the test

2. test_class_average_no_grades_equal is failing the test due to the elements of the lists in grades being strings. Even if the strings are floats in single quotes, they cannot be mathematically operated on and so there is a TypeError.
   test_class_average_many_students is failing the test because the set weights on line 60 is in the wrong order. Since the sorted() function sorts the grades in ascending order weights should be in ascending order as well. However, it's currently in descending order so it is causing an assertion error where the expected and the actual result are different.

3. test_section_average_all_grades_equal passed despite the error affecting test_class_average_no_grades_equal because for the first error, it was only localized to the second test so there was no way it could affect any others. It also passed despite the error affecting test_class_average_many_students because even though the elements in weights were out of order, that didn't matter for test_section_average_all_grades_equal since for each student, all three of their grades were the same so it didn't matter how each one would be weighted.

Part 4: Adding Noise to an Image

Complete this part in the provided `a1_part4.py` starter file. Do **not** include your solution in this file.

Part 5: Removing Noise From an Image

Implementation

Complete this part in the provided `a1_part5.py` starter file. Do **not** include your solution in this file.

Exploration

1. Because the pixels still change colour, the image starts to blur/homogenize as a given pixel will start to look more like its neighbors the more times the median filter is ran on a noise-less image.

2. A higher k value means a less chance for a given pixel to be turned white or black. So, it means that the greater k is, the less noisy the image becomes. At some point with a low enough k value, a given pixel is surrounded by enough noise such the median value in each color channel will be close to 0 or 255 since there are just that many noisy pixels around it. It's here that the median filter can't do a good job of eliminating salt and pepper noise.

3. I think that a mean filter would be worse at eliminating noise. Since the channel value for black and white lie on the ends of the range(0 and 255 respectively), they would affect a mean filter significantly more than a median filter. If only one surrounding pixel of a another given pixel was noise, it would not affect the median significantly since the majority of the other surrounding pixels are fine. However with a mean filter, that one noise pixel has a chance to greatly affect the result of the filter especially if it's a white pixel in a dark zone or a black pixel in a light zone.